Test Map Parsing

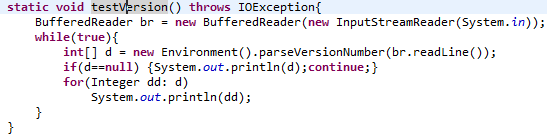
We have reviewed, debugged as well as tested the map parsing part of our game in the following key sub-sections: version control, parsing Places, parsing Paths, Parsing Lighting, and parsing Artifacts.

# Test Version Control

Game versions 1.\*, 2.\* and 3.10 are accepted. For every version, all commands are allowed.

## Test version number extraction from the GDF line in the map file

The version number “(int)a.(int)b” is stored in a int[2], with int[0] for “a” and int[1] for “b”.

Following test cases cover normal situations and possible exceptions.

|  |  |
| --- | --- |
| 3.10 //input  3 //output: version[0]  10 //output: version[1], thus v3.10 | 2.  null |
| gdf 3.10 normal game  3  10 | 2  null |
| 3.10 +3.1 -2.0  3  10 | 2. 2  Null //since there is a space inside |

## Test version control in real games

The test cases below are selected according to Equivalence Testing and Branch Testing, for exery possible situation, we select certain test cases presenting the functions we have.

* Test case: version 1.0

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\Version TestSuite\sixRooms.gdf

Game environment: GDF 1.0 The Wilderness

Input commands to wander around and find the exit.

Type HELP for instructions at any time.

Let's begin!

Result: the program accepts this game, and all commands are allowed. And for commands came up for higher versions, such as USE/GET/DROP/INVE/ASK/ANSWER, the player will get messages such as “no actor here” (for ASK) and so on.

* Test case: version 1.9, same as v1.0 actually

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\Version TestSuite\sixRooms-fakeV1.9.gdf

Game environment: GDF 1.9 The Wilderness

Input commands to wander around and find the exit.

Type HELP for instructions at any time.

Let's begin!

Result: the program accepts this game as v1.9, and all commands are allowed.

* Test case: version 2.0 - the version number is fake, its content is still under version 1.0 actually, and only one path has lock-pattern.

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\Version TestSuite\sixRooms-fakeV2.0.gdf

Game environment: GDF 2.0 The Wilderness

Input commands to wander around and find the exit.

Type HELP for instructions at any time.

Let's begin!

Result: the program accepts this game as version 2.0, and all paths, except the one with a lock-pattern in the map file, are assigned the default lock-pattern, which is unlocked. All commands are allowed to use.

* Test case: v1.0 containing a false Path

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\Version TestSuite\sixRooms-falseV1.0.gdf

Game environment: GDF 1.0 The Wilderness

ERROR: map V1.0 does not allow this many integers for path 1!

* Test case: v1.0 containing Lighting section

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\Version TestSuite\sixRooms-falseV1.1.gdf

Game environment: GDF 1.1 The Wilderness with Lighting

ERROR: map version under 2.0 doesn't allow Lighting entries!

Temporary Conclusion: for fake version numbers, we allow lower version maps with a higher fake version numbers, but not the contrary. For example, a map v1.0 containing Lighting section is not allowed. Also map v1.0 cannot have a Path with more than 3 numbers – which is the format for version 2.0 and above.

* Test case: v 3.10

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestGamePlaying\MystiCity20\_V3.10\_simplified.gdf

Game environment: GDF 3.10 MystiCity

Input commands to wander around and find the exit.

Type HELP for instructions at any time.

Let's begin!

Result: the program accepts this game, and all commands are allowed.

* Test case: version number is 3.5

…\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\Version TestSuite\sixRooms-fakeV3.5.gdf

Game environment: GDF 3.5 The Wilderness

ERROR: we currently accept only map version 1.\*, 2.\* and 3.10

Result: the program doesn’t accept this game, as we want.

## Conclusion

This is only one possible realization of the version control, and it works well now, introducing more freedom of writing map file. We have also realized another way of restricting available commands based on game version, but leave it unused.

# Test Places

All test cases for this section are stored in Git repository

\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\PlacesTestSuite

misspelledPLACE.gdf:

Parsing was stopped but using alternative filenames (as prompted to do) did not change error message except to an invalid filename.

misspelledPLACESM.gdf:

Parsing was stopped but using alternative filenames (as prompted to do) did not change error message except to an invalid filename.

PLACES\_no#.gdf:

Parsing was stopped but using alternative filenames (as prompted to do) did not change error message except to an invalid filename.

PLACES0.gdf:

Parsing was stopped, same problem as mentioned above

PLACEScomments.gdf:

Parsing was stopped and the error did not change with a new file name. The third Place was not reached because parsing was stopped during the second place.

PLACESdescriptions.gdf:

Parsing was completed. Results showed three descriptions (two extra) for the Ogre’s Lair and no descriptions for the Troll’s Bridge.

PLACESduplicateID.gdf:

Parsing was stopped. The duplicate ID was detected and the system prompted for another filename. The error message changed when a file could not be found but to nothing else.

PLACESmaxINT.gdf: - java.lang.NumberFormatException

Parsing caused an exception. The place ID number was too large for a 32 bit signed integer.

PLACESminINT.gdf: - java.lang.NumberFormatException

Parsing caused an exception. The place ID number was too large (negatively) for a 32 bit signed integer.

PLACESnoID.gdf: - java.lang.ArrayIndexOutOfBoundsException

Parsing caused an exception. The name of the place was disregarded as a useless line, and the description was added to a non-existent Place, which threw the exception

PLACESnoNAME.gdf:

Parsing was stopped but using alternative filenames (as prompted to do) did not change error message

PLACESoutOfOrder.gdf:

Parsing was finished. The alternate order for ID and name were parsed properly despite the blatant order reversal.

PLACESoverestimated.gdf:

Parsing was finished. The proposed third place was not searched for and never erroneously added.

PLACESroomID0.gdf:

Parsing was finished. The reserved room ID of 0 was assigned to the parsed place.

PLACESroomID1.gdf:

Parsing was finished. The reserved room ID of 1 was assigned to the parsed place.

PLACESunderestimated.gdf:

Parsing was finished. The extra unaccounted place was added without any error.

## Conclusion

Parsing descriptions needs to be more rigorous. Potentially for this is to make sure the last line parsed created a place or was a description. Room ID’s of 0 and 1 are allowed, but more information is needed to determine if this is a map creator problem or a parsing problem. Under or over estimating the Places follows the same rules, as the Room ID’s. Even if integers outside of the possible ranges of signed integers are not allowed, they should not cause an exception and need to be changed as well. Finally, the error detected that was not associated with places involved being unable to provide a valid filename after parsing was stopped from an invalid gdf file. The only difference noticed after the initial parsing error message is providing a incorrect filename locked the message (and state of the program) to continuously detect a non-existent file name.

Shorter version:

* Only assign descriptions if the previous line parsed was a description or a valid ID and place name line.
* A place should not be allowed to have no description (places can have no descriptions now).
* Check to make sure numbers are within integer ranges
* If parsing is stopped, allow for new files to be parse (and for a way to quit at this stage)
* More information is needed if reserved room ID’s and over or under estimating nPlaces is allowable

# Test Paths

All test cases for this section are stored in Git repository

\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\PathsTestSuite

misspelledPATH.gdf:

Parsing was completed. No paths were detected.

misspelledPATHSM.gdf:

Parsing was completed. No paths were detected.

Paths\_no#.gdf:

Parsing was completed. The paths section was detected but not having any path information did not cause an exception or stop parsing.

PATHS0.gdf:

Parsing was completed. Although 0 paths should have technically been parsed, the 1 path included was parsed.

PATHSbadDestination.gdf

Parsing was completed. The invalid path was not erroneously parsed, and was probably disregarded because it did not have enough integers.

PATHSbadDirection.gdf

Parsing was completed. Although it was intended that north as a direction would not be parsed, north was assigned the direction.

PATHSbadSource.gdf:

Parsing was stopped. The invalid Place ID was checked, and the system prompted the entry of a valid file path. But the error message only changed when a non existent file path was entered (to No environment file found) and the program could not be moved out of this state.

PATHSdecimalID.gdf: - java.lang.NumberFormatException

Parsing caused an exception. The decimal number could not be interpreted as a integer and an exception was thrown.

PATHSmaxInt.gdf - java.lang.NumberFormatException

Parsing caused an exception. The number was outside the acceptable integer ranges and could not be interpreted as an integer.

PATHSminInt.gdf - java.lang.NumberFormatException

Parsing caused an exception. The number was outside the acceptable integer ranges and could not be interpreted as an integer.

PATHSnoDirection.gdf:

Parsing was stopped. The non existence of direction for a path stopped the parsing, and the system prompted for a new file path. The error message only changed when a non existent file path was entered (to No environment file found) and the program could not be moved out of this state.

PATHSoutOfOrder.gdf:

Parsing was completed. Despite blatantly reordering the direction and source, the parsing properly detected the intended information.

PATHSoverestimated.gdf:

Parsing was completed. The parsing did not detect that the number of parsed paths was not the same as indicated to be in the file.

PATHSplacesAfter.gdf:

Parsing was stopped. The path parsing was stopped when the place ID could not be verified, and the same problem as documented above occurred when prompted to entire a new file name.

PATHStooFewInts.gdf:

Parsing was completed. The parsing disregarded the path with too few integers and continued to parse the other paths.

PATHStooManyInts.gdf:

Parsing was completed. The parsing disregarded the path with too many integers and continued to parse the other paths.

PATHSunderestimated.gdf:

Parsing was completed. The parsing did not stop when it had reached the “predetermined” number of paths in the file.

PATHSworseDirection.gdf:

Parsing was stopped. The invalid direction was detected, stopped the parsing, and prompted the demand for a new file path. But the same error as mentioned above occurred with being unable to change the state of the program to anything other than No environment file found, and then nothing else from that.

## Conclusion

The biggest omission was not storing Path ID’s. This is most likely due to a design change that originally included PlaceOut objects for the Path functionality which eventually led to the removal of the Path class. Number formatting was the only major error strictly related to sing paths, with integers outside the ranges of acceptable integers as well as decimal numbers being parsed to integers. Being unable to reenter a file name when parsing is stopped is also another error.

# Test Lighting

All test cases for this section are stored in Git repository

\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\LightingTestSuite

LIGHTINGlevel0.gdf:

Parsing was completed. The indicated light level was placed in the correct room.

LIGHTINGlevel100.gdf:

Parsing was completed. The indicated light level was placed in the correct room.

LIGHTINGlevel101.gdf:

Parsing was completed. The indicated light level was above possible values, and it was disregarded upon parsing.

LIGHTINGmaxInt.gdf - java.lang.NumberFormatException

Parsing caused an exception. Number was outside the ranges of possible 32 bit signed integer values.

LIGHTINGminInt.gdf - java.lang.NumberFormatException

Parsing caused an exception. Number was outside the ranges of possible 32 bit signed integer values.

LIGHTINGnegative.gdf:

Parsing was completed. The indicated light level was below possible values, and it was disregarded upon parsing.

LIGHTINGnonInt.gdf:

Parsing was completed. The non integer place ID number as well as light level did not cause an exception as previous non integers have, and the invalid integers were ignored upon parsing them

LIGHTINGplace0.gdf:

Parsing was completed. The indicated light level was assigned to all places, as intended by the inclusion of place 0.

LIGHTINGtooFewInts.gdf:

Parsing was completed. The light entry was disregarded because there were too few integers.

LIGHTINGtooManyInts.gdf:

Parsing was completed. The light entry was disregarded because there were too many integers.

# Test Artifacts

All test cases for this section are stored in Git repository

\CS440\Code\Text\_Game\_V3.10\TestTextGame\TestMapParsing\ArtifactTestSuite

ARTIFACTSbadPlace.gdf

Parsing was stopped. The invalid place ID was caught and the system prompted the entry of a new gdf file.

ARTIFACTSduplicateID.gdf

Parsing was stopped. The duplicate Artifact ID’s were detected and the system prompted the entry of a new gdf file.

ARTIFACTSmovable-1.gdf

Parsing was completed. Movability of the artifact was set to -1.

ARTIFACTSmovable1001.gdf

Parsing was completed. Movability of the artifact was set to 1001.

ARTIFACTSnoArtifact.gdf

Parsing was stopped. The non existent artifact for a particular description was detected and the system prompted the entry of a new gdf file.

ARTIFACTSnoDescription.gdf

Parsing was stopped. The non existent description for a particular artifact was detected and the system prompted the entry of a new gdf file.

ARTIFACTSoverestimated.gdf

Parsing was stopped. The mismatch between reported and actual artifacts was detected and the system prompted the entry of a new gdf file.

ARTIFACTStab.gdf

Parsing was completed. The tab in the name was artifact name was removed.

ARTIFACTStooFewInts.gdf

Parsing was stopped. The shortage of integers was detected and the system prompted the entry of a new gdf file.

ARTIFACTStooManyInts.gdf

Parsing was completed. The extra integer was assumed as part of the artifact name.

ARTIFACTSunderestimated.gdf

Parsing was stopped. The mismatch between reported and actual artifacts was detected and the system prompted the entry of a new gdf file.

## Conclusion

Only two code changes need to be implemented following the results. There needs to be a movability check so invalid numbers are not assigned. And names should not contain tabs.